INNOPOLIS UNIVERSITY, DEPARTMENT OF COMPUTER SCIENCE

Assignment #1: Concurrent map

Kamil Salakhiev

September 7, 2016

https://github.com/kamilsa/pp-ass1-map

1 PROBLEM DESCRIPTION

Word	Document
the	1, 2, 4
cow	1
jumped	1, 2
moon	1, 4
quick	2
fox	2, 3

Figure 1.1: Inverted index example

Inverted index is a key-value pair collection widely used in search engines to perform fast querying over a set of documents (Fig. 1.1). Key is any word that appeared in collection of documents and Value is the list of documents where that word exist. Straight-forward implementation is to use *HashMap* datastructure from Java Collections. However HashMap does not provide thread-safe properties, e.g. in order to speed up filling Inverted Index with words we may want to parallelize upload by devoting one thread per each document. The possible issue is that when multiple thread are trying to add document to the same word race condition may occur and one or several document id may be lost.
To overcome that issue we may put code segment with addition of new word-document pair into synchronized block. In spite this solution works properly, it may lead to low performance since every insertion or reading from collection locks it entirely even if those operations are applied to different buckets.

The another solution is to utilize *ConcurrentHashMap* which does not have such problem. So we can add to one bucket while reading or writing to another without locking all collection. Overall pros of using ConcurrentHashMap against HashMap with lock are the following:

1. Thread safe, though entire collection is not locked
2. Reads can happen simultaneously
3. Writes happen simultaneously if they are preformed to the different buckets

2 Experiment

To demonstrate performance of ConcurrentHashMap against HashMap the task of filling collection by words from documents was defined. To saturate collection with words the set of 510 documents with total size of 2mb was chosen. Below we can see relation between average time and the amount of threads used to hashmap saturation (Time is vertical axis and number of threads is horizontal):

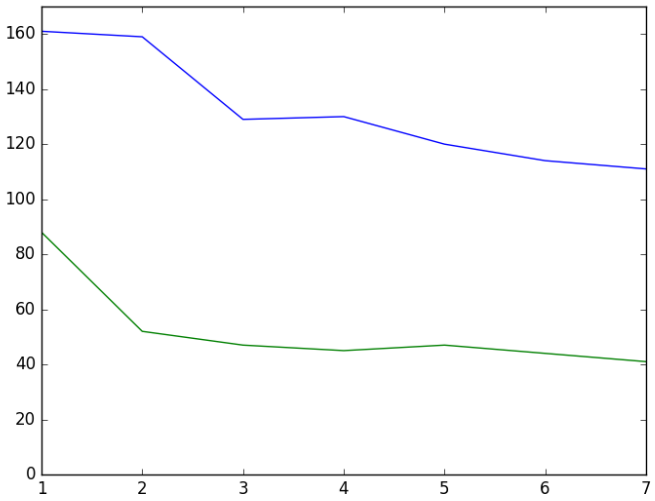

Figure 2.1: Experimental results

Obviously, concurrent implementation leads to better performance.